Neural Multi-Channel Reverse Dictionary

Nitin Kumar and **Rohan Das**
University of Colorado Boulder

Abstract

A reverse dictionary is a system that lets a user search for words based on their description or definition. Reverse Dictionaries have practical benefits in solving the "tip of the tongue' problem besides being a teaching tool for new language learners. Existing solutions for the reverse dictionary problem are unable to account for the high variability in the user descriptions and don't do well when it comes to predicting infrequently used words. Moreover, such systems use context free embeddings to encode the descriptions thus ignoring the semantic relationships between the words in the description. We propose a neural reverse dictionary model with linguistic predictors, inspired from how humans infer words from descriptions. Our model uses contextualized embeddings from RoBERTa along with multiple linguistically motivated channel predictors that aim to overcome the issues of sparsity and polysemy. We see very encouraging results that highlight the importance of contextualized embeddings and the grounding of natural language models in linguistic theory.

1 Introduction

A reverse dictionary is a task that predicts a target word based on a user's description or definition of the word. It addresses the tip-of-the-tongue problem, where people like novelists, journalists and translators often fail to retrieve a word from memory. Moreover, it is also a helpful pedagogy tool for new language learners who have limited vocabulary. We propose a neural reverse dictionary model as a solution to the "reverse dictionary" task. In addition to the practical applications mentioned above, such a task can contribute to AI interpretability, since the task involves converting human readable data to machine-readable data. Moreover, the ability to infer word embeddings from dictionary resources can help provide major insights while building solutions for low-resource languages.

This problem is challenging because it is unlikely that a user would formulate their description as exactly the one that can be found in a dictionary. Usually the user's description will consist of wording that differs from the dictionary definition. This rules out the possibility of a reverse lookup in forward dictionaries. Despite the likelihood of semantic similarities between the user's description and the actual glossary description, establishing a relationship between the two is not a simple task. Moreover it is imperative that any reverse dictionary system must be responsive since users would like to be prompted with the right words instantaneously, so as to not hamper their creative or artistic flow. The first neural models that attempted to solve this problem encoded the descriptions and target words into the same embedding space and then predicted the word that was the most similar to the description. (Hill et al., 2016). However, there are two main issues with such an approach: (1) As per Zipf's law, there are many words that are low in frequency. This sparsity issue prevents the learning of good embeddings for such words. (2) Such models do not address the issue of polysemy since they use static word embeddings and thus fail to account for different senses of the same word.

We propose a neural reverse dictionary model inspired from Zhang et al., 2020 which takes into account the cognitive process involved in inferring the target word from its description. Channel predictors are used to extract knowledge about morphemes, sememes, POS tags and the categorical nature of the words in the description. These act as a guide to help predict the word closest to the description and filters out the words whose embeddings are close to the target word embedding but are semantically different. These channel predictors address the polysemy issue. Our proposed model uses pre-trained RoBERTa (Liu et al., 2019) to encode the description in order to fix the sparsity issue. These contextualized word embeddings

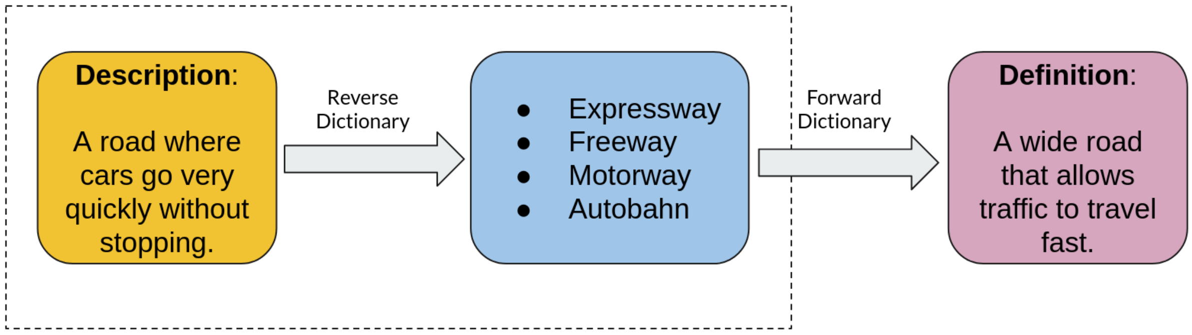
Description:
A road where cars go very quickly without stopping.
Reverse Dictionary
Expressway
Freeway
Motorway
Autobahn
Forward Dictionary
Definition:
A wide road that allows traffic to travel fast.

Figure 1: Reverse Dictionary

are used to obtain sentence level encodings of the description. Further, the feature specific predictors are added to the next layer which extract the semantic information from the description and guide the model toward the target word embedding.

2 Related Work

Most of the existing solutions to the reverse dictionary problem are grounded in information retrieval (IR) methods. Such systems establish relationships between dictionary definitions and words and the final task is reduced to that of generating a ranked list of words that are semantically similar to the definitions. At a high-level such an IR system consists of three primary components: 1. a query formulation, 2. a query expansion and 3. a semantic similarity function.

Query Formulation: Them query formulation component is responsible to formulating a query from the user's description which acts as the input to a reverse dictionary system. Bilac et al., 2004 used tf-idf and boolean expressions to build vector representation of the query, to be matched against the corresponding vector representation of the words. For short definitions such representations tend to suffer from a large number of null components. El-Kahlout and Oflazer, 2004 and Shaw et al., 2013 employed standard preprocessing methods like tokenizaiton and stop word removal for the user descriptions prior to the boolean query formulation step. This method however doesn't account for the relative importance of words in the description.

Query Expansion: This component aims to address the issue of vocabulary mismatch between the query words and the actual dictionary definitions. Bilac et al., 2004 opted to heuristically expand the queries during database preparation. However, since users are likely to use synonyms, hypernyms and hyponyms as opposed to the exact dictionary definition, the lexically related terms should also encompass the expansion terms. El-Kahlout and Oflazer, 2004 only took into account synonyms while Shaw et al., 2013 expanded the queries starting with synonyms, followed by hypernyms and hyponyms based on the number of required target words. All of these methods however don't do a good job at word sense disambiguation.

Semantic Function Similarity: This component is responsible for selecting definitions that have semantic similarity with the description and generate a ranked list of target words. Most works opted for the more common cosine-similarity function for this task while others opted for direct matching/occurrence of the words. Although computationally simple, none of these methods solve the polysemy issue or take into account the importance of word order.

Graph based solutions usually employ semantic search algorithms on lexical graphs to find candidate words matching the user description. Dutoit and Nugues, 2002 proposed a two-step algorithm built on top of the Integral Dictionary lexical network which involved word extraction followed by ranking based on a semantic distance measure. Thorat and Choudhari, 2016 on the other hand constructed a lexical graph using dictionary definitions. The semantic search algorithm that they used involved evolving the graph in relation to each word in the description, followed by computing similarity based on depth and finally ranking the target words. The drawback to this approach is that the

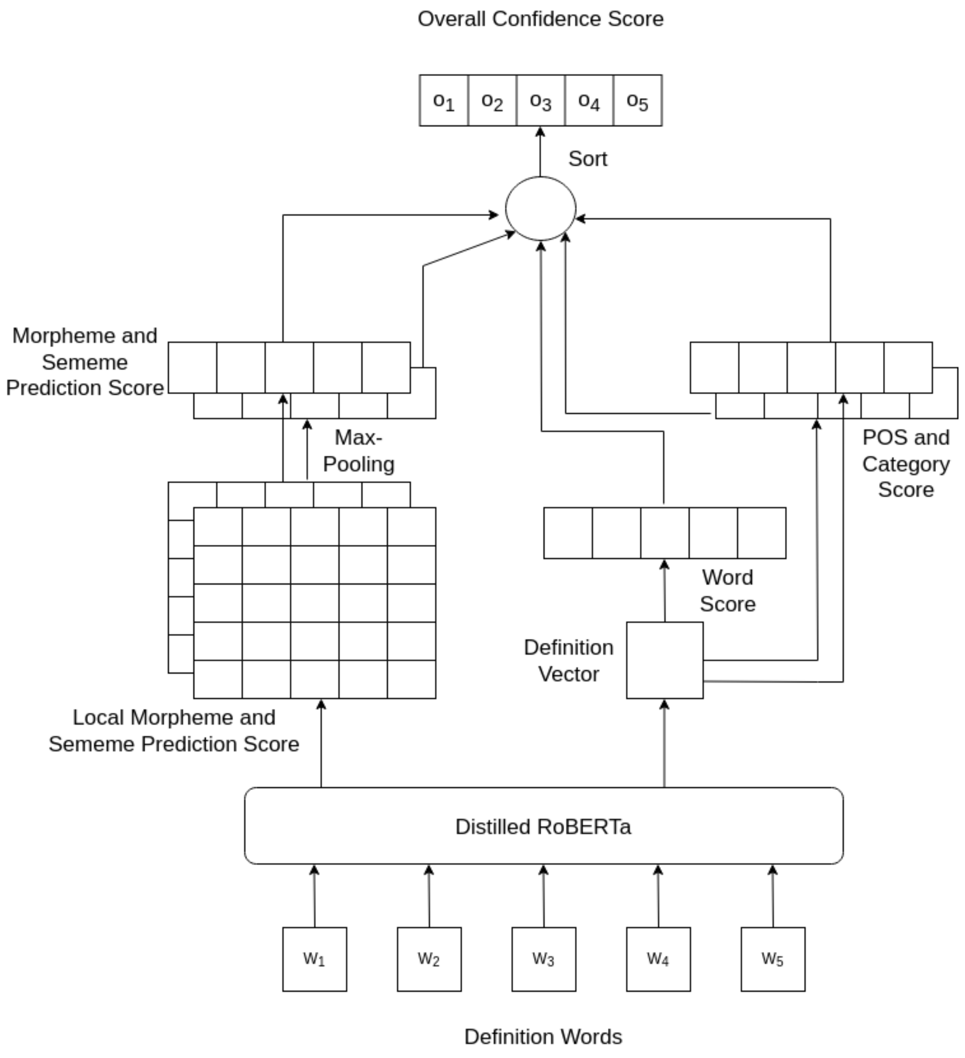

Figure 2: System Architecture

assumption of semantic similarity between words at equal depth is not necessarily true always. Such graph based architectures also work only on small lexicons of less than 3000 words.

Hill et al., 2016 experimented with both LSTMs and a linear bag of words model along with pretrained word embeddings. Both approaches showed similar results. In order to remove irrelevant results, Morinaga and Yamaguchi, 2018 added category inference while Kartsaklis et al., 2018 proposed a system that learnt word representations from a WordNet graph and used a LSTM with incorporated multi-sense to encode the user descriptions. Zhang et al., 2020 employed a multi-channel predictor system to incorporate information about morphemes, sememes and part-of-speech tags to mimic how humans infer words from descriptions. All of these approaches used static embeddings like word2Vec and thus failed to capture the context in which words in the input query related to each other.

We propose a multi-channel neural model that uses pre-trained RoBERTa to embed the input queries. The multi-channel predictors help the model to identify characteristics of candidate words and thus mimic the human inference process. Representing the input queries using contextualized embeddings help the model to achieve better semantic understanding of the user description.

3 Methods

In this section, we describe the overall system architecture of our proposed model. We first give an overview of the workflow and then do a deep dive into the specifics of our model.

3.1 Workflow

The user description is encoded using contextualized embeddings and fed as input to the neural multi-channel model, which calculates a confidence score for each candidate word in the vocabulary. After obtaining confidence scores, all candidates are sorted in the order of descending confidence scores. All candidate words that appear in the user description are excluded since they are unlikely to be the target word. The ranked list of candidate words are returned.

3.2 Neural multi-channel Reverse Dictionary Model

The confidence score for each candidate word for a given user description is computed as follows:

Word Score: The user description is encoded using pre-trained RoBERTa as the sentence encoder. This encodes the user description into a query vector and captures the contextual importance of all the words in the sentence. The sentence vector is fed to a two layer perceptron, to obtain a sentence embedding mapped in the space of the individual word embeddings. The final confidence score is calculated using a cosine similarity score between the sentence vector and each candidate word's embedding.

Part of Speech Score: A prediction score is computed for each POS tag by feeding the sentence vector into a single-layer feed forward network. The final POS score of the candidate word is just the summation of the scores of all its POS tags. This solves the issue of returning words with tags that don't match the user description.

Category Score: A word category score helps in eliminating words that are semantically related but not similar from the candidate list. It is calculated in a manner similar to the POS score.

Morpheme Score: Morphemes of a word have a local semantic relationship with its definition. For this reason we include a morpheme predictor. The hidden states are fed to a single-layer feed forward network and a local score is obtained. Subsequently we do a max-pooling over all the morpheme scores to obtain a score for each morpheme. Finally, a the morpheme score of a candidate word is calculated as the sum of the prediction score of all its morphemes.

Sememe Score: Sememes, the minimum semantic unit in a language, help accurately depict the word meaning. The sememe score of a candidate word is calculated in a manner similar to the morpheme score.

4 Experimental Design

4.1 Dataset

For the training data, we use the datasets introduced in Hill et al., 2016. This dataset consists of 100K words and 900K word-definition pairs. We extract the morphemes from the words using Morfessor (Smit et al., 2014). Category information and Parts of Speech tags are obtained from WordNet (Miller, 1995), Further, We use OpenHowNet (Qi et al., 2019) to retrieve sememes.

We use three different test sets for evaluation: (1) A **Seen Definition** set that consists of 500 word-definition pairs drawn from the training set. This test set helps evaluate the model's ability to recall examples it has already seen during training. (2) An **Unseen Definition** set that also consists of 500 word-definition pairs, and (3) A **Description** set which acts as a benchmark set consisting of 200 word-hand crafted description pairs.

4.2 Model Configuration

We trained the following models:

4.2.1 Baseline

For our baseline model we use the model proposed by Zhang et al., 2020 which encoded the description using a word2Vec embeddings passed through a Bi-directional LSTM (BiLSTM) encoder.

4.2.2 System 1

For our first experimental system we trained the exact model proposed by Zhang et al., 2020 but we used GloVe embeddings instead of Word2vec. Unlike Word2vec, GloVe is a prediction based model that takes into account the co-occurrence statistics of words across the entire corpus.

4.2.3 System 2

Next, we trained the Zhang et al., 2020 model with fastText embeddings. fastText is known to perform better compared to Word2vec on smaller training sets and is able to leverage subword information. We also trained another version of this model, but without the channel predictors for comparison.

Model	Seen Definition	Unseen Definition	Description
	Accuracy@1/10/100	Accuracy@1/10/100	Accuracy@1/10/100
word2Vec + BiLSTM	.20/.46/.69	.09/.28/.60	.32/.64/.89
GloVe + BiLSTM	.17/.39/.65	.06/.22/.53	.30/.64/.89
fastText + BiLSTM	.26/.55/.81	.14/.43/.73	.35/.73/.87
fastText + BiLSTM w/o channel predictors	.31/.63/.86	.08/.36/.69	.31/.65/.85
RoBERTa w/o fine-tuning	.02/.09/23	.01/.07/.17	.09/.30/.62
RoBERTa + fine-tuning + LayerNorm	.25/.52/.69	.24/.50/.66	.20/.46/.74

Table 1: Evaluation Results on the Accuracy@1/10/100 metric

Model	Seen Definition		Unseen Definition		Description	
	Median Rank	Rank Variance	Median Rank	Rank Variance	Median Rank	Rank Variance
word2Vec + BiLSTM	13	307.41	55.5	356.62	3	221.35
GloVe + BiLSTM	25	356.24	82.5	403.72	3	179.70
fastText + BiLSTM	6	249.67	16.5	311.38	1.5	218.95
fastText + BiLSTM w/o channel predictors	3	200.91	21.5	323.9	3	266.55
RoBERTa w/o fine-tuning	1000	426.78	1000	411.30	43	361.50
RoBERTa + fine-tuning + LayerNorm	8	360.52	10.5	356.80	12	366.33

Table 2: Evaluation Results on the Median Rank and Rank Variance metrics

4.2.4 System 3

For our third system we train a model that involved encoding the word descriptions using a distilled RoBERTa encoder model and augment it using multi-channel linguistic predictors. Using RoBERTa will enable the model to generate a contextualized description vector which should do a better job at capturing semantic relationships. Here the weights of the RoBERTa encoder were frozen during training. We use a single-layer perceptron to map the sentence vector into space of the word embeddings.

4.2.5 System 4

For our final system we train our proposed model that involved encoding the word descriptions using a distilled RoBERTa encoder model and augment it using multi-channel linguistic predictors. We fine-tune RoBERTa during the training process and add a second single-layer perceptron along with layer normalization.

4.3 Hyperparameters and Training

For the models with context-free membeddings, we used 300 dimensional embeddings and hidden unit size of 300 as well. For the RoBERTa based models we used 768 dimensional embeddings and a hidden unit size of 768. We used a Cross Entropy Loss function and ReLU as the activation function. We also used 'Adam' as optimizer with an initial learning rate of 0.001 and a batch size of 512. All of our models were trained on a single Nvidia V100 GPU on Google Colab and University of Colorado Boulder's Research Computing Cluster. The baseline model and systems 1 and 2 were trained for 20 epochs, for an average training time of 1 hour. The final RoBERTa based model was trained for 15 epochs for a total training time of over 18 hours.

4.4 Evaluation Metrics

For every test example, all models produce a ranked list of candidate words. We evaluate this ranked list on the basis of three statistics which are the most widely used metrics for the reverse dictionary task as proposed by Hill et al., 2016: (1) the **median rank** of the correct candidate word over the whole test set where a lower number is better. When ranked, it denotes how close the candidate words are from the ground truth words (2) the proportion of training examples in which the candidate word appears in the top 1/10/100 in the ranked list. This is denoted by **Accuracy@1/10/100** and a higher number is better, and (3) the **rank variance** of the correct candidate word over the entire test set.

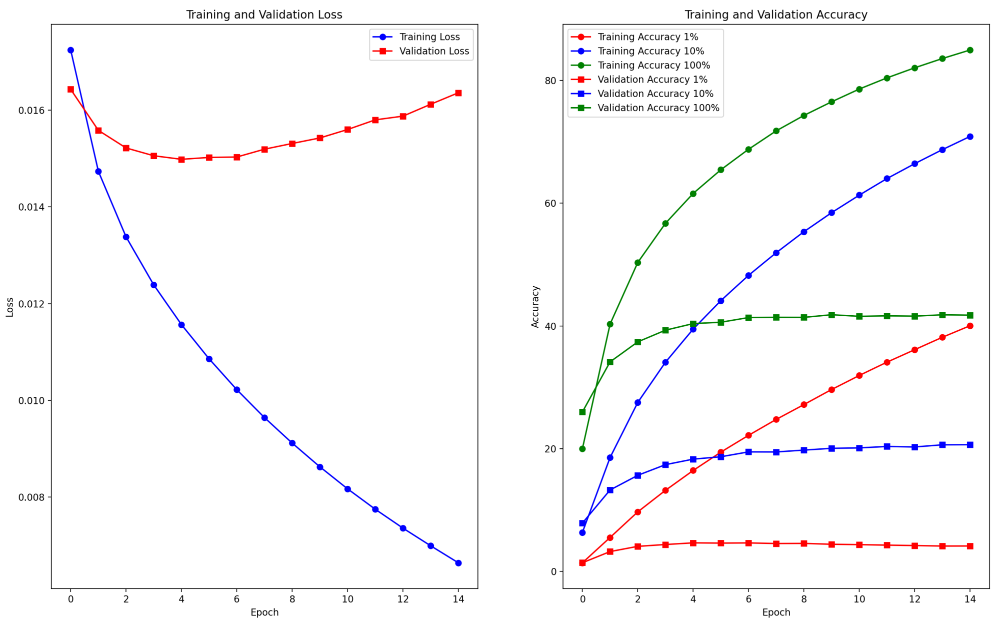

Figure 3: Loss and Accuracy for fastText + BiLSTM

Again, a lower number is desirable. It indicates the variability of the correct candidate word as well the robustness of the model to output the correct candidate word.

5 Experimental Results

We successfully replicated the results of the baseline model from the Zhang et al., 2020 paper. The baseline doesn't perform very well on the unseen definition set, especially on the Top 1 and Top 10 rankings. In fact across all models that use context-free embeddings, we see significantly better performance on the description set. We believe this is due to the nature of the words used in the user descriptions. It is likely that user descriptions tend to consist of synonyms or words that are very similar to the candidate word, whereas the definition may not necessarily consist of words that are similar to the target word. Given this similarity aspect it appears that the context-free models tend to predict words in the same embedding space and thus tend to do better on the description set compared to the unseen definition set.

GloVe seems to perform slightly poorer compared to Word2vec. Since, Word2vec tends to leverage local context more compared to GloVe (which focuses more on global word co-occurrence counts), the Word2vec based model is able to better capture local semantic relationships that enables it to predict the target word more accurately.

The model with fastText embeddings performed significantly better than both Word2vec and GloVe on the unseen definition set. This model also has the overall best performance on the description set beating the baseline by quite a good margin on both Top 1 and Top 10 accuracy. Given that fastText embeddings can leverage subword information, it has the ability to generate embeddings for out of vocabulary words. Along linguistic lines, fastText is also known to incorporate morphological information, which we believe played a significant role in its performance on the unseen definition set as it allows the model to generalize better on the definition words that may not necessarily appear in the same embedding space. The median ranks of 16.5 and 1.5 on the unseen definition and description sets are also an indicator of the robust behaviour of this model. We also trained a fastText model with all the channel predictors turned off and we see a 6 point drop in performance on average across all test sets. This reinforces the importance of linguistic predictors in the reverse dictionary task.

RoBERTa performed very poorly when the weights were frozen (i.e. without fine-tuning) both during the training as well as testing phases denoting that the model was severely underfitting.

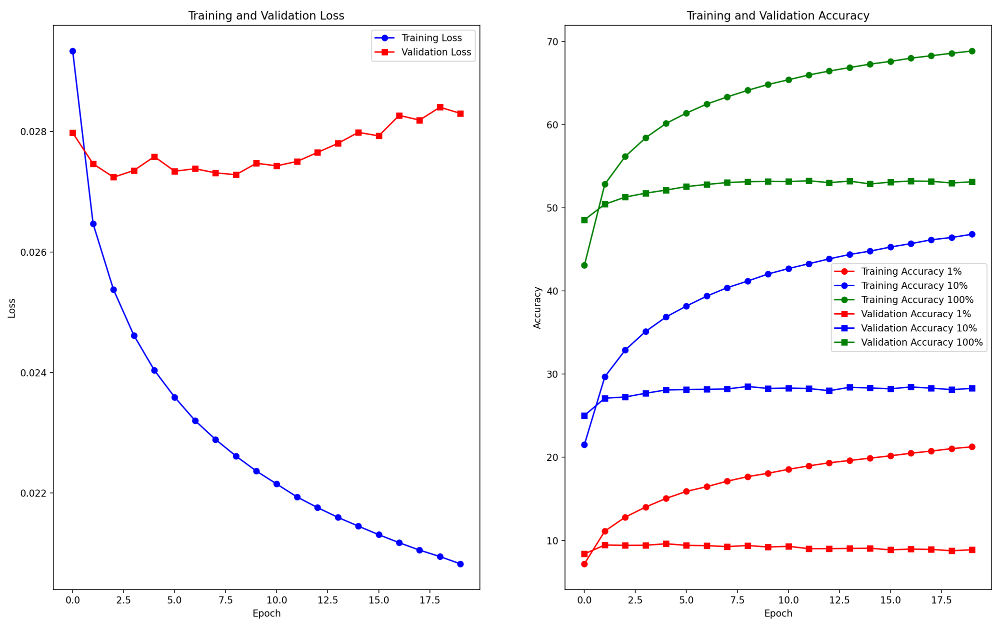

Figure 4: Loss and Accuracy for RoBERTa + fine-tuning + LayerNorm

It seemed to need more hidden layers so that the complexity of 768 dimensional input embeddings could be mapped well with the output dimension of 75101 which is the total number of words in the vocabulary. Looking at the results for RoBERTa without fine-tuning, we realized the need for increasing the model complexity. Instead of freezing the RoBERTa model weights, we chose to fine-tune it and we also added an additional fully connected layer and layer normalization to the overall architecture. It performed better than our baseline model and thus was able to model the complexity needed to map with the output. Although it did not perform better than fastText on the seen definition set, it still showed the best results on the unseen definition set for top 1 and top 10 words. This observation is also supported by the median rank of 10.5 that the model achieves on the unseen definition set which is the best among all the models. It shows how well RoBERTa is able to generalize due to the use of contextualized embeddings. We also observed that the loss for the model does not converge even after 15 epochs and 18 hours of training. Thus, further hyperparameter tuning (learning rate improvements and addition of hidden layers with layer normalization) is needed to make the model converge faster and potentially generalize better to the description set.

Validation loss for all models slightly decreased at first but then increased, even though the validation accuracy keeps on increasing. This hints at potential overfitting, but the increase in the test accuracy undermines this proposition. Other possible reason could be mis-calibration (i.e. a mismatch between confidence and accuracy of the model) which could be due to the use of cross entropy loss. Focal loss (Mukhoti et al., 2020) could be used to calibrate the loss better which minimizes regularized KL (Kullback-Leibler) divergence unlike cross entropy loss which minimizes KL divergence.

It is important to note that the RoBERTa model bucks the overall trend by performing poorly on the description set compared to the unseen definition set. A possible reason for this could be the need for more training but this claim needs more empirical validation. In the future, we also intend to train and validate whether the model is able to generalize well on out of domain words especially from scientific and medical domains. We also hope to conduct a deeper qualitative analysis of our models to find out the influence of factors like number of word senses, frequency of words in the vocabulary and the length of the input description.

6 Conclusion

In this paper, we present a neural reverse dictionary model supported by multi-channel linguistic predictors. Our RoBERTa based model showed that contextualized embeddings play an important role in generalizing well over different test sets compared to context-free embeddings. Reverse Dictionary models not only have important practical applications in language pedagogy but it also helps language models to better interpret and represent the meaning of phrases and sentences. We also observed the strong influence that linguistic predictors like morphemes and sememes have on the model, that lead to substantially better predictions. We strongly encourage the larger NLP community to focus more on grounding language models in linguistic theory as a viable alternative to building large language models that are computationally intensive and difficult to deploy.

Although it is likely to assume that dictionaries would be free from gender bias, there has been evidence that suggests this may not be the case. For example, the Oxford Dictionary has been known to use male pronouns when describing or defining words like *doctorate*, *scientist* and *smart*. We need to be careful when training our models on dictionary definitions with inherent bias so as to not allow gender bias to unknowingly creep into our models.

References

Slaven Bilac, Wataru Watanabe, Taiichi Hashimoto, Takenobu Tokunaga, and Hozumi Tanaka. 2004. Dictionary search based on the target word description. In *Proceedings of the Tenth Annual Meeting of the Association for NLP (NLP2004)*, page 556–559.

Dominique Dutoit and Pierre Nugues. 2002. A lexical database and an algorithm to find words from definitions. In *ECAI*.

Ilknur Durgar El-Kahlout and Kemal Oflazer. 2004. Use of wordnet for retrieving words from their meanings.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30.

Dimitri Kartsaklis, Mohammad Taher Pilehvar, and Nigel Collier. 2018. Mapping text to knowledge graph entities using multi-sense lstms. In *EMNLP*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Yuya Morinaga and Kazunori Yamaguchi. 2018. Improvement of reverse dictionary by tuning word vectors and category inference. In *ICIST*.

Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip H. S. Torr, and Puneet Kumar Dokania. 2020. Calibrating deep neural networks using focal loss. *ArXiv*, abs/2002.09437.

Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Qiang Dong, Maosong Sun, and Zhendong Dong. 2019. Openhownet: An open sememe-based lexical knowledge base. *ArXiv*, abs/1901.09957.

Ryan B. Shaw, Anindya Datta, Debra E. VanderMeer, and Kaushik Dutta. 2013. Building a scalable database-driven reverse dictionary. *IEEE Transactions on Knowledge and Data Engineering*, 25:528–540.

Peter Smit, Sami Virpioja, Stig-Arne Grönroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. pages 21–24.

Sushrut Thorat and Varad Choudhari. 2016. Implementing a reverse dictionary, based on word definitions, using a node-graph architecture. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2797–2806, Osaka, Japan. The COLING 2016 Organizing Committee.

Lei Zhang, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2020. Multi-channel reverse dictionary model. In *AAAI*.